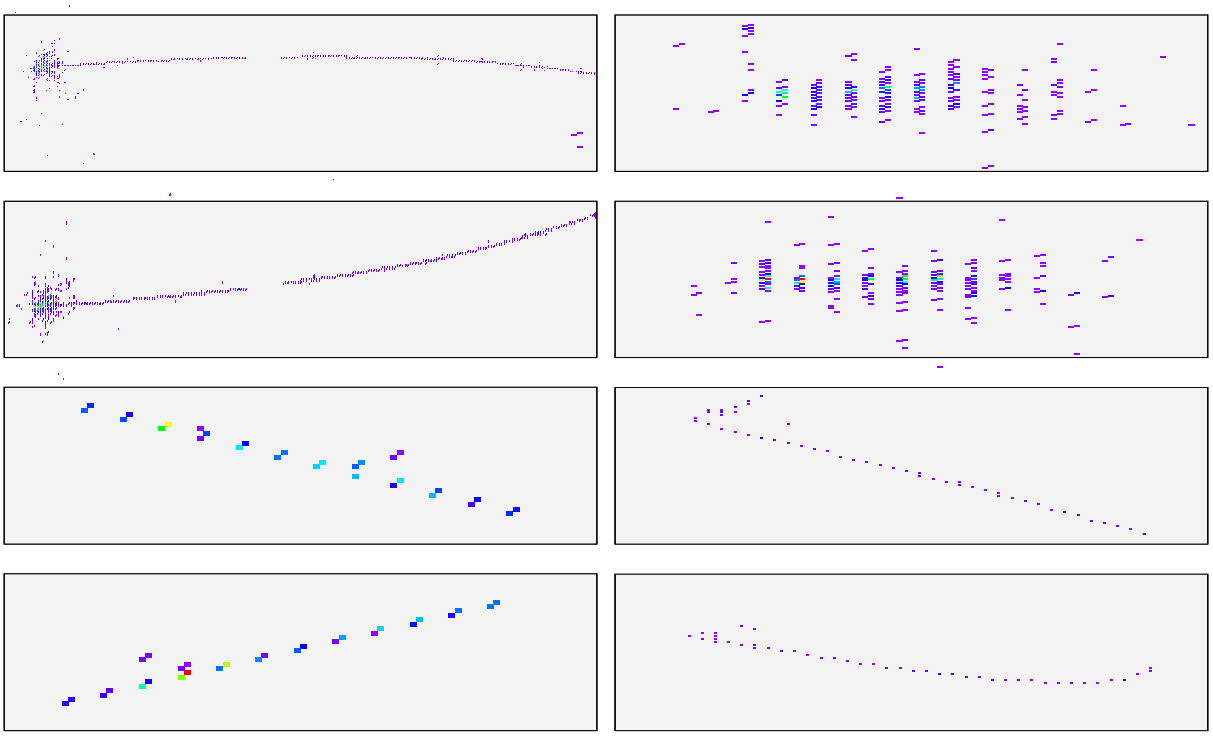
MIDAD: Minos Interactive Data Analysis and Display



- Minos - mini site report.
- Guiding design.
- Main Classes of the Framework.
- Screenshot of current display.
- ROOT: Problems and Suggestions.
  - CINT, Rt and TG GUI classes.

http://minos.phy.bnl.gov/~bviren/talks/

MINOS - mini site report

Hardware Three somewhat similar detectors:

- Near detector (ND) - at Fermilab.
- Far detector (FD) - in northern Minnesota.
- Calibration detector (CalDet) - portable - presently in CERN test beam

Software Time Scale :

- Now→Sept, 2001, usable prototype ready for CalDet and FD (cosmic muons).
- 2001→2003, reassess, redesign, rewrite as necessary.

Current Software Strategy : Heavy use of ROOT. Little attempt to isolate most of MINOS framework from ROOT.

Comments from Sept, 2000 MINOS software review: (paraphrased from several external reviewers)

"[In your software infrastructure you should be using C++ constructs like templates, exceptions, RTTI, STL containers/iterators, standard library, etc., instead of letting ROOT usage dictate which subset of the C++ standard language you work in.]"

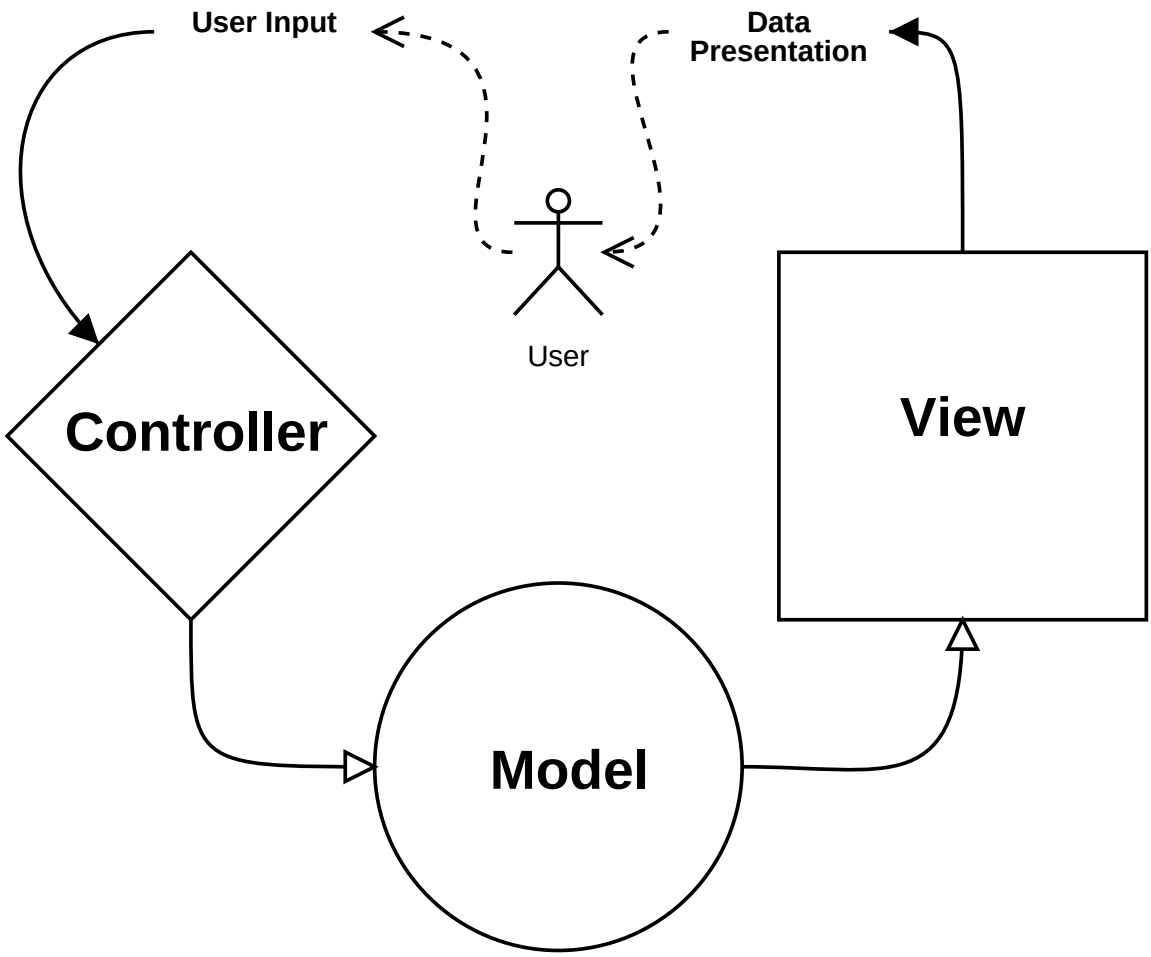
Minos software future:

- How ROOT evolves is likely to determine whether or how we need to moderate our use of ROOT to meet these criticisms.

Major MIDAD Design Criteria

- Present multiple graphical and non graphical representations of data and MC objects.
- Display multiple objects, eg. raw data, reconstructed tracks/showers, MC truth.
- Ability to easily add custom displays.
- Ability to "mark-up" a display with text/arrows.
- Allow display of standard and custom user histograms.
- Allow user drawing code.
- Ability to drive reconstruction with interactively altered data.

Model-View-Controller pattern



Model Encapsulates data, signals when data is modified, exports interface to modify data.

View Passive display of data, told by model if data is changed.

Controller Modifies data via Model's interface.

Pattern-Oriented Software Arch, A Sys. of Patterns, Buschmann, *et.al.*

Main MIDAD Framework classes

Model A TQObject derived base class for encapsulating a data/MC object (hits, track, showers, etc). Maintains meta-data (eg. selected hits). Emits an Rt signal when meta data changes.

Viewable A TNamed and TQObject derived base class for graphical representations of particular Model. Emits Rt signals on mouse enter/exit/click.

View A TPad derived class to which Viewables are added. Accepts only Viewables which implement a particular view type (projection). Emits Rt signals in response to selections.

DisplayBase A TGMainFrame derived base class for collecting one or more Views and other GUI elements. Installs Viewables in to its Views.

Supportive Management Classes

ViewableProxy templated class instantiated as a static object at link time, one for each concrete Viewable type. Used to register the concrete Viewable type and provide creator functions.

ViewableReg Singleton holding registered ViewableProxys, responsible for creating Viewables.

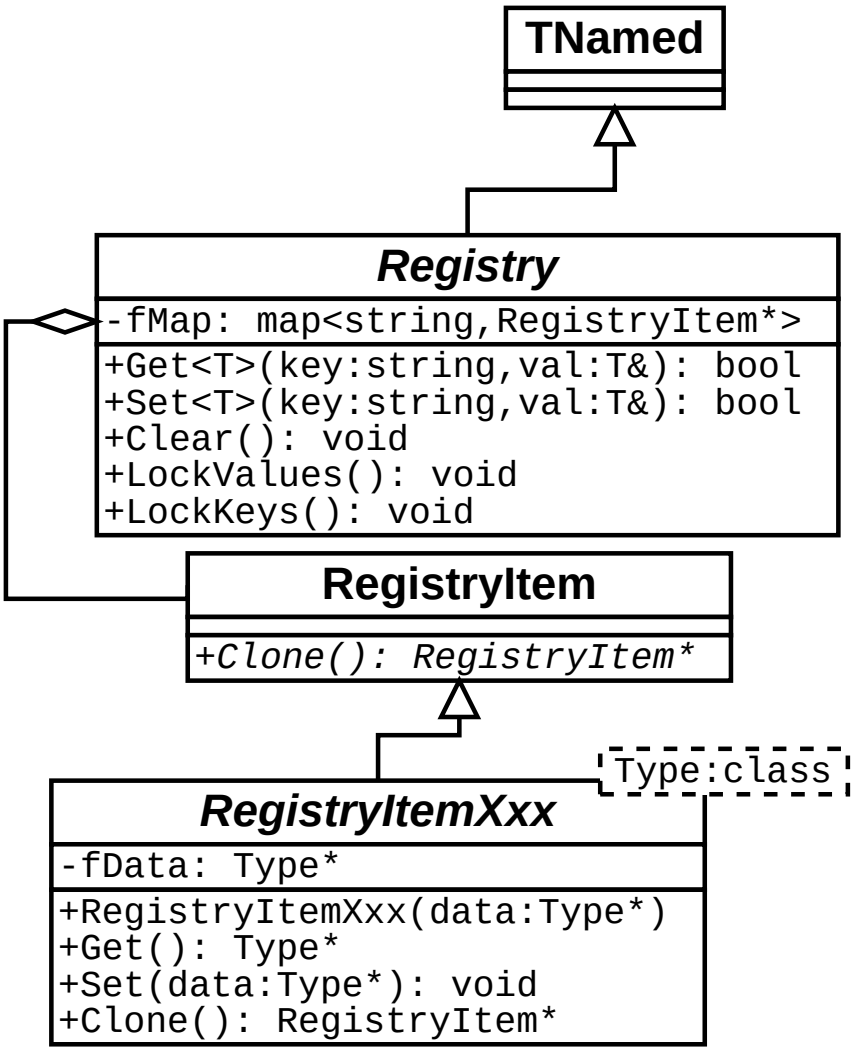
ModelProxy Same as ViewableProxy, but for concrete Models.

ModelHistory Holds collection of one type of Model. Emits an Rt signal when Models are added/removed. Allows for undo/redo. Creates Models by name.

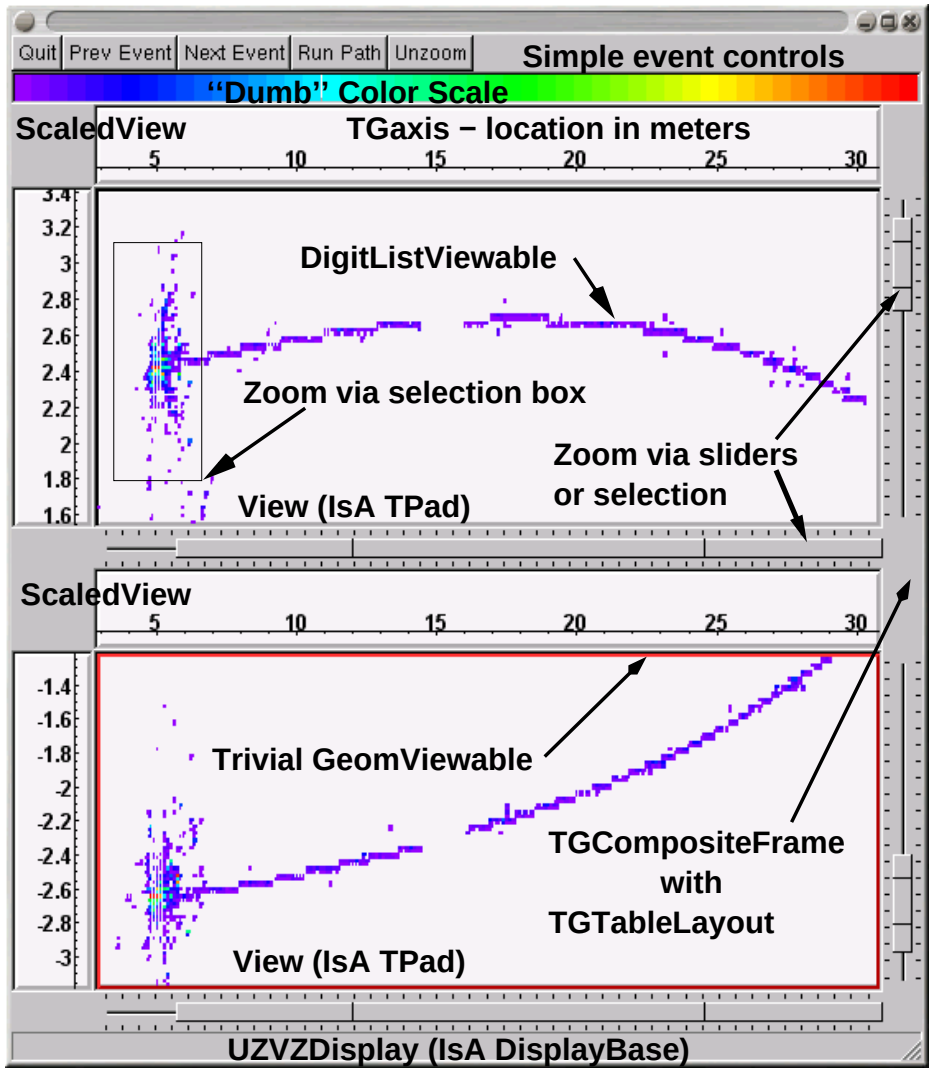
ModelPool Like ViewableReg but creates/leases ModelHistorys.

Registry

- Used to configure Viewables (and other MINOS objects).
- Heterogenous, type safe mapping from strings typed keys to arbitrary typed values.
- Traverses ROOT I/O.



Example Screenshot



First and very simple MIDAD display.

Coding with ROOT, Problems and Suggestions

- ROOT has much to be praised for.
  - Comprehensive, becoming well documented.
  - Strong and knowledgeable community testing, debugging and improving it.
  - Rene, Fons and Masaharu are quick to help, address problems and integrate new features.
- However, it's more useful to criticize it than to praise.
- The following is based on difficulties developing MIDAD, particularly in the area of trying to use templates with ROOT.

CINT

CINT is very cool. For rapid development under ROOT, it is very useful. Given the complexity of interpreting C++, it is understandable that CINT is not 100% perfect.

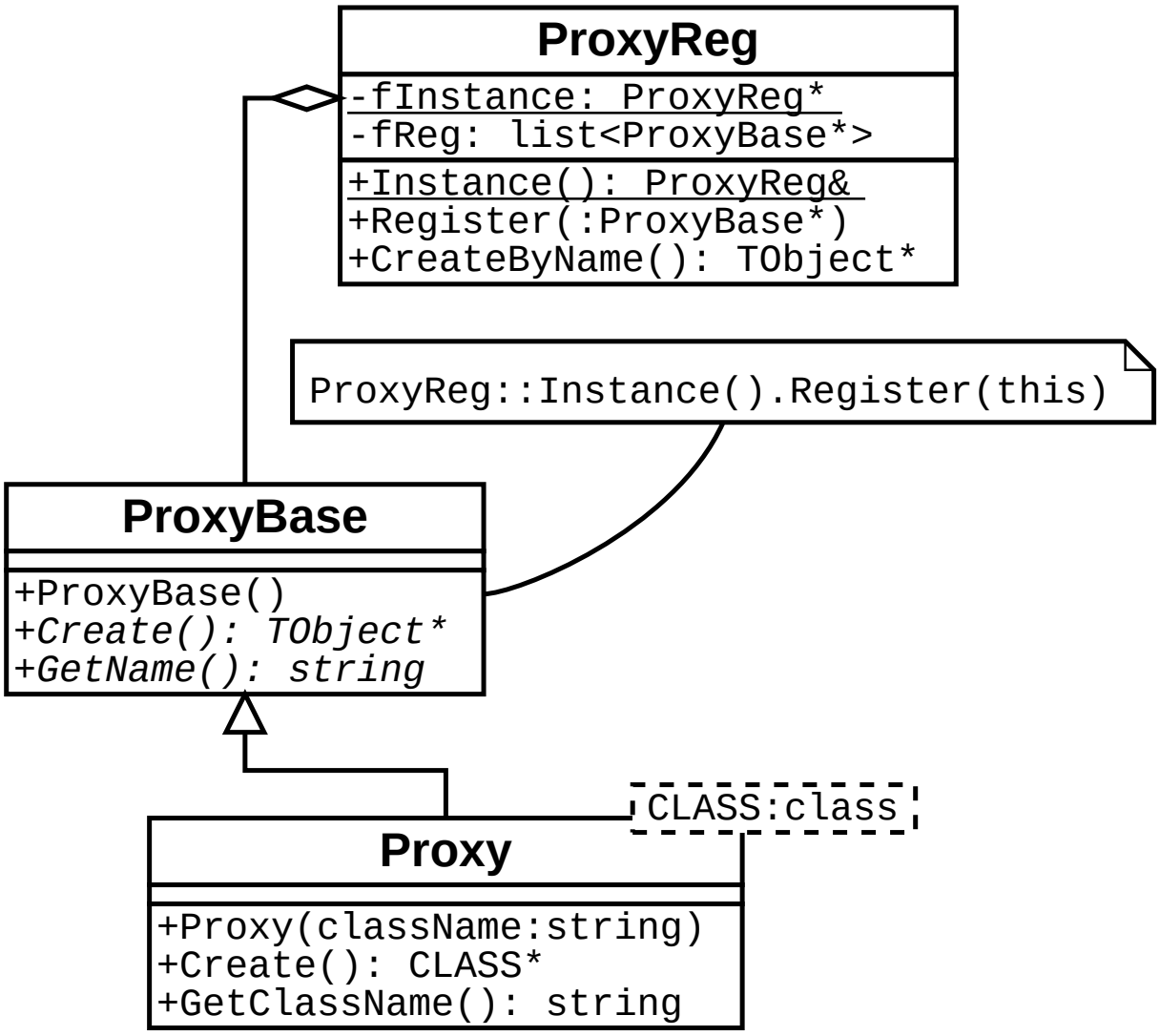
CINT inside compiled code leads to hidden/confusing dependencies (TApplication ↔ TRootGuiFactory). Uncertain. Why is it necessary?.

Modern C++ (STL, templates, exceptions, namespaces). Partial CINT support (and improving), ROOT containers, TString redundant with STL. ROOT predates STL, but will it adapt/adopt? Wish: full template support in CINT, ROOT subclasses STL containers and strings.

Linkdef.h ordering is critical w/ templates. Document rules / likely problems helps. Solution?

CPP macros (ClassDef{,T,TT,...,T2,TT2,...,Nested} & ClassImp{,T,TT,...,Q,Nested}) not scalable nor clear.
Can templates + inheritance help? → Proxy.

Templated Proxy




```
// MyRootClass.cxx (IsA TObject)
#include "MyRootClass.h"
// ...
static Proxy<MyRootClass>
        gsMyRootClassProxy("MyRootClass");
// ...
```

CINT as both interpreter and MOP.

CINT is used for both ROOT's extension & scripting language as well as providing ROOT with a Meta-Object Protocol (MOP) to extend the C++ language (Dictionary, Serialization).

Interesting alternative: **OpenC++**.

- Compile time MOP.
- Extended C++ language (less so than CINT). No CPP macros nor cmds in comments.
- Emits standard C++ code.
- MOP allows for object persistence (I/O).
- Allows optimizations like:
  `Matrix m = m1+m2+m3;`
- http://www.csg.is.titech.ac.jp/~chiba/openc++.html
- See "A Metaobject Protocol For C++", S. Chiba (available from above).

Rt Signal/Slot mechanism

Some praise:

- Signal/slot mechanism is very useful, almost indispensable, part of a class library.
- Rt is a good implementation and works well with what it has as a base (CINT).
- Sig/slots bridge the compiled/interpreted gap.

Some criticism:

- Run-time and string based nature leads to programming errors & reliance on CINT in compiled code.
- Multiple or user types must be passed by casting pointers to Long_t.
- Binding args at connection time done via `sprintf()`'ing address into char buffer.
- Quiet failures (Valeriy: "existing checks unused - no interpreted class Dictionaries").
- No templated signals or slots (CINT limitation).

Suggestion:

- Consider libsigc++ (libsigc.sourceforge.net).

TG GUI Classes and TGuiFactory

Praise: Cross platform. Integrated (a good thing?). Some support for implementation in other GUIs (Denis Bertini's QtRoot, my embryonic GtkRoot)

Some Criticism:

- Documentation slim.
- Difficult/confusing design. Not as high quality as "professional" GUI TKs (eg. Qt, Gtk).
- Need more widgets implemented.
- Tie between TG imp and rest of Root is too tight.
- GUI's best handled by GUI experts.

Suggestions (fairly obvious):

- Docs for use, extension and implementation of GUI.
- Keep accepting contributed TG widgets (blatant plug: TGTableLayout. Think HTML/LaTeX tables holding widgets in heterogenous columns/rows).
- Continue to be open to the adoption/support of QtRoot, GtkRoot and future XxxRoot → helps keep ROOT modular and broadens ROOT's appeal and thus outside contribution.

Summary

- An event display framework for MINOS has been (and continues to be) developed.
- Much of this development was made possible by the quality and quantity of ROOT's feature set, documentation, community and developer support.
- Various perceived problems were given and possible suggestions were offered in the hopes of making ROOT even better.
- Underlining these suggestions is a wish for ROOT to work towards continuing to better its support for "modern" C++ features, particularly templates.